Available online at www.sciencedirect.com
SCIENCE DIRECT®

Theoretical Computer Science 356 (2006) 356–373

Theoretical Computer Science

www.elsevier.com/locate/tcs

Concurrent nets: A study of prefixing in process calculi

Emmanuel Beffara*, François Maurel

Équipe PPS, Université Paris 7 & CNRS, France

Abstract

We introduce the calculus of concurrent nets as an extension of the fusion calculus in which usual prefixing is replaced by arbitrary monotonic guards. Then we use this formalism to describe the prefixing policy of standard calculi as a particular form of communication. By developing a graphical syntax, we sharpen the geometric intuition and finally we provide an encoding of these guards as causality in the prefix-free fragment, in the spirit of the encoding of the fusion calculus into solos by Laneve and Victor, proving that communication by fusion is expressive enough to implement arbitrary monotonic guards.

Keywords: π-calculus; Fusion calculus; Graphical syntax; Prefixing

1. Introduction

The π-calculus [10] has generated a wide range of calculi on the search for both a simplification of the syntax and a widening of the expressiveness. Fu's χ-calculus [2], Parrow and Victor's fusion calculus [12] and Gardner and Wischik's explicit fusions [3] are important examples where name substitution is replaced by unification, which makes the calculus simpler, more symmetric and yet more expressive. Most models for concurrent and mobile computation are geometric in nature, and even term calculi have a strong spatial intuition. Indeed, every process calculus comes with a handful of structural rules for commutation and scoping that define appropriate notions of locality. This geometric flavour of term calculi led to the proposal of several graphical syntaxes for existing calculi, like π-nets [9] or solo diagrams [7], and to the introduction of new purely graphical calculi, like in Milner's recent work on bigraphs [5].

In a sense, the evolution from name substitution in π-calculus to fusion corresponds to the evolution from syntactical communication to a more geometric one; however, the sequentiality imposed by prefixing remains very syntactical, since it is directly inspired by CCS and synchronisation trees. Motivated by the search for a more general form of prefixing, we introduce and study the calculus of concurrent nets as a similar evolution towards a geometrical formulation of sequentiality constraints. Actions in a process get associated with semaphores that indicate when these actions have been performed, and subsequently prefixing is replaced by the use of guards that are monotonic functions of those semaphores, resulting in a form of sequentiality constraints that is reminiscent of enabling in event structures. We define a graphical syntax for this calculus, in which guards appear as a new form of communication between actions.

* Corresponding author. PPS, Case 7014, 2 Place Jussieu, 75251 Paris Cedex 05, France.
E-mail address: beffara@pps.jussieu.fr (E. Beffara).

0304-3975/$ - see front matter © 2006 Elsevier B.V. All rights reserved.
doi:10.1016/j.tcs.2006.02.009

With the calculus of solos [8], Laneve and Victor simplify the fusion calculus by removing prefixing and they show by means of encodings that no expressive power is lost. The idea of using the expressiveness of fusion for expressing scheduling constraints can be generalised to our framework of arbitrary monotonic guards.

The first contribution of this paper is to show how concurrent nets extend existing calculi, and notably the π-calculus and the fusion calculus. The characterisation of these sub-calculi yields a classification of the forms of sequentiality in use in process calculi using natural geometric arguments.

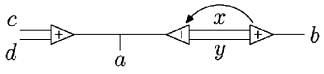
Our second contribution shows the completeness of the expressive power of communication by fusion with respect to monotonic scheduling, by means of encodings of the arbitrary monotonic guards of concurrent nets into pure communications, using a new and more geometric approach suggested by our graphical syntax.

In Section 2, the syntax and semantics of the calculus of concurrent nets are introduced, both as a term calculus and as a graph reduction formalism. In Section 3, we characterise various existing calculi as restrictions on guarding and scoping. In Section 4, we develop two encodings of monotonic guards into communications. In Section 5, we study how guards, considered as a form of communication, interact with replication and recursion.

2. Definitions

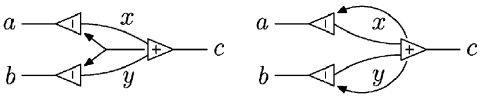
2.1. Introductory examples

Informally, a concurrent net is a web of input and output actions related by channels. Each action has a principal port (the subject) and a set of auxiliary ports (the objects). For instance, the process $\bar{a}(cd)|a(xy).\,\bar{b}(xy)$ in π-calculus is represented as



Formally, the set of channels here is $\mathcal{C} = \{a, b, c, d, x, y\}$, but only a, b, c, d are considered public, which is represented by the fact that they have dangling edges while x and y have none. The actions $\bar{a}(cd)$ on the left and $a(xy)$ in the middle form a redex. The reduction of this redex will remove both actions and connect c with x and d with y (Fig. 1). The arrow means that the third action $\bar{b}(xy)$ is prefixed by $a(xy)$, i.e. it will not be reduced as long as $a(xy)$ is present.

We generalise this prefixing in two ways. Consider the following examples:



In both examples, we have two receptions $a(x)$ and $b(y)$ and one emission $\bar{c}(xy)$. In the process on the left, the two-headed arrow means that the emission is prefixed by both receptions, i.e. $\bar{c}(xy)$ will be blocked until both $a(x)$ and $b(y)$ have been reduced, but these may happen in any order. This cannot be expressed directly in π-calculus, but some calculi (e.g. the join calculus [1]) do provide this kind of synchronisation. In the process on the right, the two disjoint arrows mean that $\bar{c}(xy)$ will be able to act as soon as either $a(x)$ or $b(y)$ is consumed. To our knowledge, this too cannot be expressed directly in other calculi. Note that, if $a(x)$ is consumed, y may be communicated before anyone writes on b. This phenomenon is typical of fusion calculi.

2.2. Syntax and semantics

We assume a countable set $\mathcal{C}$ of channel names and a disjoint set $\mathcal{L}$ of location names. The elements of $\mathcal{L}$ are used to name occurrences of actions, as detailed below. We write $\tilde{x}$ to represent a finite sequence of channel names $x_1 \ldots x_{|\tilde{x}|}$.

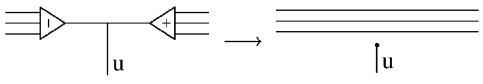


Fig. 1. Reduction of a redex.

Definition 1. $\mathcal{C}$-terms are defined by the following grammar:

actions	$\alpha := \bar{u}(\tilde{x}) \mid u(\tilde{x})$,
guards	$\pi := 0 \mid 1 \mid \ell \mid \pi + \pi \mid \pi\pi$,
terms	$P := \mathbf{0} \mid (P\|P) \mid (\nu x)P \mid (\nu \ell)P \mid \ell : \alpha \mid \langle \pi \rangle P$,

where u, x, x_i range over channel names and ℓ ranges over location names.

- The α-*equivalence* on $\mathcal{C}$-terms is generated by the renaming of bound names. A channel or location name x is bound by the closest surrounding (νx).
- A location ℓ is *defined* in P if some $\ell : \alpha$ occurs in P with ℓ unbound. We denote by $\mathrm{loc}(P)$, the set of locations defined in P.
- We require that each location be defined at most once in any subterm (this condition on terms will be preserved by reduction).
- A term P is *prefix-closed* if any location name that occurs in its prefixes is element of $\mathrm{loc}(P)$.

The actions $\bar{u}(\tilde{x})$ and $u(\tilde{x})$ correspond to the emission and reception of some sequence of channels $\tilde{x}$ on a channel u. Guards are either blocked (0), enabled (1), simple (a location ℓ), disjunctive $(\pi_1 + \pi_2)$ or conjunctive $(\pi_1\pi_2)$. A prefixing $\langle \ell \rangle P$ means that P is blocked until the action $\ell : \alpha$ is performed. When this happens, the name ℓ is replaced by 1 in every prefix. Subsequently, a process $\langle \pi \rangle P$ is blocked until the prefix π is reduced to 1 by the replacement of some ℓ by 1 and the application of structural rules. The other constructions are standard: $\mathbf{0}$ is the inactive process, $P_1|P_2$ represents two processes in parallel and $(\nu x)P$ represents the process P with a local name x.

We use the notation α for $(\nu \ell)(\ell : \alpha)$, i.e. when the location ℓ is not used in any prefix. The notation $u^{\varepsilon}(\tilde{x})$ refers to an action with arbitrary polarity, where ε is $+$ for the action $\bar{u}(\tilde{x})$ and $-$ for $u(\tilde{x})$.

Example 2. The process

$$\text{shared\_continuation}(a, b, P) := (\nu \ell_1)(\nu \ell_2)\big(\ell_1 : a() \mid \ell_2 : b() \mid \langle \ell_1 + \ell_2 \rangle P\big)$$

is a typical process that does not exist primitively in usual calculi such as fusion calculus. The process P in shared_continuation(a, b, P) is enabled by the unblocking of ℓ_1 or ℓ_2 or both (the location ℓ_1 is unblocked when the occurrence of $a()$ in $\ell_1 : a()$ is used in reduction and similarly for ℓ_2 and $b()$). Even if it may be encoded in fusion calculus, as shown in Section 4, such a mechanism is not primitive. For instance, a similar process in fusion calculus would be $Q = (\nu f)(a().\bar{f}() \mid b().\bar{f}() \mid f().P)$ which has the intended meaning: when a or b has been performed, the flag f is released and P becomes available. The methodological difference is quite subtle: in this example, the encoding works because there is only one $\bar{f}()$ that can interact (and the other possible $\bar{f}()$ is not used and can be garbage collected) whereas in shared_continuation(a, b, P), both ℓ_1 and ℓ_2 enable P and no garbage collection rule is necessary.

Example 3. Another more involved example is the process

$$P = (\nu \ell_a \ell_b \ell_c)\big(\, \ell_a : a_1() \mid \ell_b : b_1() \mid \ell_c : c_1() \mid \langle \ell_a \rangle \bar{a}_2() \mid \langle \ell_b \rangle \bar{b}_2() \mid \langle \ell_c \rangle \bar{c}_2() \mid \langle \ell_a \ell_b \rangle \bar{d}() \mid \langle \ell_a + \ell_b \ell_c \rangle \bar{e}() \,\big).$$

The process P listens on three ports a_1, b_1 and c_1. For each of them it answers, respectively, on ports a_2, b_2 and c_2. Furthermore, when both $a_1()$ and $b_1()$ have been fired, P sends $\bar{d}()$, and when $a_1()$ or both $b_1()$ and $c_1()$ have been fired $\bar{e}()$ is sent. In fusion calculus, a similar process could be

$$\begin{aligned} Q = (\nu f_a f_{b1} f_{b2} f_c g)\big(\, & a_1().\big(\bar{f}_a() \mid \bar{g}() \mid \bar{a}_2()\big) \mid\ b_1().\big(\bar{f}_{b1}() \mid \bar{f}_{b2}() \mid \bar{b}_2()\big) \mid c_1().\big(\bar{f}_c() \mid \bar{c}_2()\big) \\ & \mid f_a().f_{b1}().\bar{d}() \mid f_{b2}().f_c().\bar{g}() \mid g().\bar{e}() \,\big) \end{aligned}$$

but this is more complex: one must be cautious when programming such a process to check that the flags f_a, f_{b1}, f_{b2}, f_c and g are sufficient for the intended purpose. Moreover, the encoding is too sequential: in the encoding, we write $f_a().f_{b1}().\bar{d}()$ but could as well write $f_{b1}().f_a().\bar{d}()$ in this case. This asymmetry prevents an easy and natural understanding of such a process.

These processes exemplify the situation where one has simple processes in $\mathcal{C}$-terms and tricky encodings in fusion. This situation is typical as shown by the encodings (in solos) developed in Section 4. Therefore, we see $\mathcal{C}$-terms as a plain process calculus with important properties but also as some kind of macro language giving new *design patterns* for process calculi such as π or solos.

Definition 4. The structural equivalence on terms is the smallest congruence $\equiv$ containing α-equivalence and such that
- the set of terms is a commutative monoid with $|$ as the composition and $\mathbf{0}$ as the neutral element;
- let Π be the set of guards, then $(\Pi, 0, +)$ and $(\Pi, 1, .)$ are commutative monoids, mutually distributive, with $1+x \equiv 1$ and $0\,x \equiv 0$;
- prefixing obeys the following rules, where ℓ does not occur in π:

$$\begin{aligned}
\langle 0\rangle P &\equiv \mathbf{0} && \text{nullity},\\
\langle 1\rangle P &\equiv P && \text{neutrality},\\
\langle \pi_1\rangle\langle \pi_2\rangle P &\equiv \langle \pi_1\pi_2\rangle P && \text{composition},\\
\langle \pi\rangle(P_1|P_2) &\equiv \langle \pi\rangle P_1 \mid \langle \pi\rangle P_2 && \text{distribution},\\
\langle \pi\rangle(\boldsymbol{\nu} x)P &\equiv (\boldsymbol{\nu} x)\langle \pi\rangle P && \text{channel scoping},\\
\langle \pi\rangle(\boldsymbol{\nu} \ell)P &\equiv (\boldsymbol{\nu} \ell)\langle \pi\rangle P && \text{location scoping, when } \ell \notin \mathrm{fv}(\pi);
\end{aligned}$$

- channel and location scoping obeys the following standard equivalence rules, where z is not free in P, and n is neither defined nor used in P:

$$\begin{aligned}
(\boldsymbol{\nu} x)(\boldsymbol{\nu} y)P &\equiv (\boldsymbol{\nu} y)(\boldsymbol{\nu} x)P, \quad (\boldsymbol{\nu} z)P \equiv P \quad P \mid (\boldsymbol{\nu} z)Q \equiv (\boldsymbol{\nu} z)(P|Q),\\
(\boldsymbol{\nu} \ell)(\boldsymbol{\nu} m)P &\equiv (\boldsymbol{\nu} m)(\boldsymbol{\nu} \ell)P, \quad (\boldsymbol{\nu} n)P \equiv P, \quad P \mid (\boldsymbol{\nu} n)Q \equiv (\boldsymbol{\nu} n)(P|Q).
\end{aligned}$$

The operational semantics of $\mathcal{C}$-terms is defined as a labelled transition system (LTS). The choice of an LTS instead of a simple reduction system comes from fusion effects and prefix updates. When a process P reduces into P', the process $P|Q$ reduces into $P'|Q'$ where Q' is Q with some unified variables and some reduced prefixes (the reduction in P may have unblocked some locations that appear as prefixes in Q). Hence, for compositionality, we use an LTS which can properly deal with parallel composition.

Transitions are labelled either (φ, L) or $((\boldsymbol{\nu}\tilde{x})\alpha, L)$ where φ is an equivalence over $\mathcal{C}$, L is a subset of $\mathcal{L}$, $\tilde{x}$ is a subset of $\mathcal{C}$ and α is an action. (φ, L) means that a unification φ is performed, and $((\boldsymbol{\nu}\tilde{x})\alpha, L)$ means that the action α is fired and the scopes of some variables $\tilde{x}$ are extended. In both cases, L is the set of locations that are unblocked (the actions in the locations L have been fired).

Definition 5. We write $\{\tilde{x} = \tilde{y}\}$ to denote the smallest equivalence that unifies $\tilde{x}$ with $\tilde{y}$, $x \notin \varphi$ if the equivalence class of x is $\{x\}$, $\varphi \setminus x$ for $\varphi \cap (\mathcal{C} \setminus \{x\})^2 \cup \{(x, x)\}$, and $[1/L]$ for the substitution of each name in L by 1 in prefixes. A substitution σ *implements* a relation φ if σ is idempotent and $x\ \varphi\ y$ iff $\sigma(x) = \sigma(y)$. The transition rules for $\mathcal{C}$-terms, up to structural equivalence, are given in Table 1.

When an action $\ell : \alpha$ is performed, the name ℓ is replaced by 1 in the rest of the process. The name ℓ can be interpreted as that of a global variable (or a semaphore) with a monotonic value, initially set to 0, which gets the value 1 on activation of α. Then a prefix is a monotonic combination of semaphores.

Definition 6. Bisimilarity on $\mathcal{C}$-terms is defined as follows:
- A bisimulation is a symmetric binary relation $\mathcal{S}$ over processes such that $P\mathcal{S}Q$ implies that for all transition $P \xrightarrow{e,L} P'$ there is a process Q' such that $Q \xrightarrow{e,L} Q'$ and $P'\mathcal{S}Q'$.
- A relation $\mathcal{S}$ is *stable* if it is closed under arbitrary name substitution and if $P\mathcal{S}Q$ implies $P[1/L]\mathcal{S}Q[1/L]$ for all $L \subset \mathcal{L} \setminus (\mathrm{loc}(P) \cup \mathrm{loc}(Q))$.

Two processes are (stably) bisimilar if they are related by a (stable) bisimulation.

Table 1
Labelled transition system for $\mathcal{C}$-terms

$$\frac{}{\ell : \alpha \xrightarrow{\alpha,\{\ell\}} \mathbf{0}} \qquad \frac{P_1 \xrightarrow{(\nu\tilde{x}_1)\bar{a}(\tilde{y}_1),L_1} P_1' \qquad P_2 \xrightarrow{(\nu\tilde{x}_2)a(\tilde{y}_2),L_2} P_2' \qquad |\tilde{y}_1| = |\tilde{y}_2|}{P_1|P_2 \xrightarrow{\{\tilde{y}_1=\tilde{y}_2\}\setminus\tilde{x}_1\tilde{x}_2,L_1\cup L_2} (\boldsymbol{\nu}\tilde{x}_1\tilde{x}_2)(P_1'|P_2')\sigma[1/L_1, L_2]},$$

where σ implements $\{\tilde{y}_1 = \tilde{y}_2\}$ and $z \notin \tilde{x}_1\tilde{x}_2 \Rightarrow \sigma(z) \notin \tilde{x}_1\tilde{x}_2$

$$\frac{P \xrightarrow{(\nu\tilde{x})a^{\varepsilon}(\tilde{y}),L} P' \qquad z \notin a\tilde{y}}{(\boldsymbol{\nu}z)P \xrightarrow{(\nu\tilde{x})a^{\varepsilon}(\tilde{y}),L} (\boldsymbol{\nu}z)P'} \qquad \frac{P \xrightarrow{\varphi,L} P' \qquad z \notin \varphi}{(\boldsymbol{\nu}z)P \xrightarrow{\varphi,L} (\boldsymbol{\nu}z)P'}$$

$$\frac{P \xrightarrow{(\nu\tilde{x})a^{\varepsilon}(\tilde{y}),L} P' \qquad z \in \tilde{y},\ z \neq a}{(\boldsymbol{\nu}z)P \xrightarrow{(\nu z\tilde{x})a^{\varepsilon}(\tilde{y}),L} P'} \qquad \frac{P \xrightarrow{\varphi,L} P' \qquad z\ \varphi\ y,\ z \neq y}{(\boldsymbol{\nu}z)P \xrightarrow{\varphi\setminus\{z\},L} P'[y/z]}$$

$$\frac{P \xrightarrow{(\nu\tilde{x})a^{\varepsilon}(\tilde{y}),L} P'}{(\boldsymbol{\nu}\ell)P \xrightarrow{(\nu\tilde{x})a^{\varepsilon}(\tilde{y}),L\setminus\{\ell\}} (\boldsymbol{\nu}\ell)P'} \qquad \frac{P \xrightarrow{\varphi,L} P'}{(\boldsymbol{\nu}\ell)P \xrightarrow{\varphi,L\setminus\{\ell\}} (\boldsymbol{\nu}\ell)P'}$$

$$\frac{P \xrightarrow{(\nu\tilde{x})\alpha,L} P'}{P|Q \xrightarrow{(\nu\tilde{x})\alpha,L} P'|Q[1/L]} \qquad \frac{P \xrightarrow{\varphi,L} P'}{P|Q \xrightarrow{\varphi,L} P'|Q[1/L]}$$

Without the stability clause, our definition is standard bisimulation. Stable bisimilarity is more pertinent, because bisimilarity is not preserved under the effects of the context. For instance, let

$$P = \ell_1 : \bar{a}(), \quad Q = \ell_1 : \bar{a}() \mid \langle m\rangle\ell_2 : a(), \quad R = m : \bar{c}() \mid c().$$

P and Q are bisimilar since they have the same transition labelled $(\bar{a}(), \{\ell_1\})$, leading to the trivially bisimilar processes $\mathbf{0}$ and $\langle m\rangle\ell_2 : a()$, respectively. However, $P|R$ and $Q|R$ are not bisimilar: both have one transition labelled (id, $\{m\}$), which leads to P in the case of $P|R$ and to $\ell_1 : \bar{a}() \mid \ell_2 : a()$ in the case of $Q|R$, and these reducts cannot be bisimilar since the former has no transition while the latter has one. So a stable bisimulation is a bisimulation that is preserved under the effects that the context may produce. Note that the condition above restricts L to be composed of locations that are *not defined* in P or Q: these locations may occur in the guards in P and Q, and they may be substituted by 1 because of transitions in the context, as illustrated in the previous example.

Our encodings of guards in Section 4 are not bisimulations; they require the weaker equivalence of *barbed* bisimulation:

Definition 7. Barbed bisimilarity on $\mathcal{C}$-terms is defined as follows:

- A process P has a *barb* on (u, L) if there is a transition $P \xrightarrow{(\nu\tilde{x})u^{\varepsilon}(\tilde{y}),L} P'$ for some $\tilde{x}$, $\tilde{y}$, ε, and P'. We denote it $P\downarrow(u, L)$.
- A barbed bisimulation is a symmetric binary relation $\mathcal{S}$ over processes such that $P\mathcal{S}Q$ implies
  - for all u, L, if $P\downarrow(u, L)$ then $Q\downarrow(u, L)$,
  - for any transition $P \xrightarrow{\varphi,L} P'$ there is a Q' such that $Q \xrightarrow{\varphi,L} Q'$ and $P'\sigma\mathcal{S}Q'\sigma$ for some substitution σ that implements φ.

Two processes are (stably) barb-bisimilar if they are related by a (stable) barbed bisimulation.

It also makes sense to define associated notions of weak bisimulation, in which only observable transitions are considered. Observability here refers both to name fusion and location freeing, i.e. a transition $\xrightarrow{\varphi,L}$ is observable as soon as φ is not the identity or L is not empty.

Definition 8. Weak bisimilarity on $\mathcal{C}$-terms is defined as follows:

- The τ-reduction relation $\rightarrow$ is defined as $P \rightarrow P'$ iff $P \xrightarrow{\mathrm{id},\emptyset} P'$ where id stands for the identity relation $\{(x, x) \mid x \in \mathcal{C}\}$. The relation $\rightarrow^*$ is the reflexive transitive closure of $\rightarrow$.

- A weak bisimulation is a symmetric relation $\mathcal{S}$ over processes such that $P\mathcal{S}Q$ implies that for any $P \xrightarrow{e,L} P'$ with $e \neq (\mathrm{id}, \emptyset)$, there is a Q' such that $Q \to^* \xrightarrow{e,L} Q'$ and $P'\mathcal{S}Q'$.

Two processes are (stably) weakly bisimilar if they are related by a (stable) weak bisimulation.

Definition 9. A weak barbed bisimulation is a symmetric relation $\mathcal{S}$ over processes such that $P\mathcal{S}Q$ implies

- for all u, L, if $P\downarrow(u, L)$ then $Q \to^* \downarrow(u, L)$,
- for any $P \xrightarrow{\varphi,L} P'$ with $(\varphi, L) \neq (\mathrm{id}, \emptyset)$, there is a Q' such that $Q \to^* \xrightarrow{\varphi,L} Q'$ and $P'\sigma\mathcal{S}Q'\sigma$ for some substitution σ that implements φ.

Two processes are (stably) weakly barb-bisimilar if they are related by a (stable) weak barbed bisimulation.

2.3. Graphical syntax

The calculus of $\mathcal{C}$-terms has a large number of structural rules to define appropriate notions of locality and scope in processes. The following canonical form property yields a graphical formulation that avoids the need for such rules.

Proposition 10. *Any prefix-closed $\mathcal{C}$-term P is structurally equivalent to a term with the following shape, where the product stands for parallel composition*:

$$P \equiv (\nu\tilde{w})(\nu\tilde{m}) \prod_{i=1}^{n} \langle \pi_i \rangle \ell_i : u_i^{e_i}(\tilde{x}_i) \quad \textit{with} \quad \pi_i \equiv \sum_{j=1}^{p_i} \ell_{i,j,1} \cdots \ell_{i,j,q_{i,j}}$$

for some $n \geqslant 0$, $p_i \geqslant 0$ and $q_{i,j} \geqslant 0$, where the $\ell_{i,j,k}$ are elements of $\{\ell_i \mid 1 \leqslant i \leqslant n\}$. The sets $\tilde{w}$ and $\tilde{m}$ represent, respectively, private channels and locations. Such a formulation is called an enumeration of P.

Proof. By scope extrusion, all binders may be moved to the top level of the syntax tree, and the distribution and composition rules for prefixes lead to the expression of P as a composition of elementary guarded actions. The standard form of guards is obtained by distributivity of conjunction over disjunction. □

Hence, a process can be described as a set of locations, each with an associated action and prefix. Actions are built on a set of channel names, some of which are bound. Unbound channels form a set called the interface. The prefix of an action is either 1 or a disjunction of non-empty sets of locations. Prefixes define a relation: the enabling relation between non-empty sets of locations (i.e. occurrences of actions) and actions, in the spirit of event structures. This leads to the following algebraic definition, where $\mathcal{C}^*$ stands for the set of finite sequences over $\mathcal{C}$ and $\mathcal{P}_0(\mathcal{A})$ is the set of non-empty subsets of $\mathcal{A}$ (the non-emptiness condition is justified after Definition 12).

Definition 11. A concurrent net consists of

- a set $\mathcal{C}$ of *channels*,
- a subset $\mathcal{I}$ of $\mathcal{C}$ called the *interface*,
- a set $\mathcal{A}$ of *actions* labelled by elements of $\{+, -\} \times \mathcal{C} \times \mathcal{C}^*$,
- an *enabling* relation $\vdash$ between $\mathcal{P}_0(\mathcal{A})$ and $\mathcal{A}$.

In the sequel, channels are ranged over by Latin letters and actions are ranged over by Greek letters. A positive action $(+, u, \tilde{x})$ is written as $\bar{u}(\tilde{x})$ and a negative action $(-, u, \tilde{x})$ is written as $u(\tilde{x})$. In such an action, u is the *principal* channel and the elements of $\tilde{x}$ are the *auxiliary* channels.

Fig. 2 shows the graphical conventions we use to represent concurrent nets: the channels are the edges in a hypergraph over actions, the channels in the interface are those with dangling edges. Actions are represented by triangles with the polarity in the middle, the principal channel (the subject) is connected to a vertex of the triangle and the auxiliary channels are connected to the opposite side. By convention, the auxiliary ports of negative actions are ordered from left to right (when looking from the principal channel) while those of positive actions are ordered from right to left, which leads to cleaner figures. The enabling relation is represented by arrows: for each element $\beta_1, \ldots, \beta_n \vdash \alpha$, we draw a multi-headed arrow from α to each of the β_i. Different elements of $\vdash$ are represented by disjoint arrows. The arrows represent guards, so an action is enabled when there is no arrow leaving its node, and communication only occurs between enabled actions.

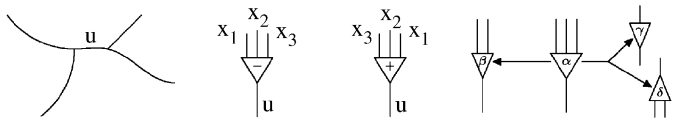


Fig. 2. Graphical syntax for concurrent nets.

Definition 12. Let $P = (\mathcal{C}, \mathcal{I}, \mathcal{A}, \vdash)$ be a concurrent net.

- An action α is *enabled* if there is no set X such that $X \vdash \alpha$.
- A *redex* is a pair $\{\alpha, \beta\}$ of enabled actions of opposite polarities with the same subject and arity.
- P reduces along the redex $\{\alpha, \beta\}$ into the net P' obtained by removing α and β from the quotient of P by $\{\tilde{x} = \tilde{y}\}$. If there is an arrow $X \vdash \gamma$ in P with $X \subseteq \{\alpha, \beta\}$, then any arrow $Y \vdash \gamma$ is removed in P'. Otherwise, any arrow $X \vdash \gamma$ in P is replaced by $X \setminus \{\alpha, \beta\} \vdash \gamma$.

If an arrow $\emptyset \vdash \alpha$ appears in the reduction, then α gets enabled and all arrows $Y \vdash \alpha$ are removed, which corresponds to the axiom $1 + x = 1$. This axiom is precisely what is needed in Definition 13 to make the graphical formulation equivalent to $\mathcal{C}$-terms. An equivalent approach would be to allow empty sets on the left of $\vdash$ and to define that α is enabled when $\emptyset \vdash \alpha$.

As illustrated by Fig. 1, the reduction of a redex consists in removing the redex and connecting the auxiliary channels of the positive action with those of the negative action. The principal channel of the actions (u in the figure) is still present in the net, minus two actions. Any arrow that points to an action in the redex is removed, which possibly enables other actions.

Definition 13. The equivalence $\equiv$ over concurrent nets is the smallest equivalence such that a net with an arrow $X \vdash \alpha$ is equivalent to the same net plus an arrow $Y \vdash \alpha$ for any set of actions Y such that $X \subseteq Y$.

This equivalence is characterised by the operation that removes every arrow $Y \vdash \alpha$ for which there exists $X \vdash \alpha$ with $X \subseteq Y$. Two nets are structurally equivalent if their images by this transformation are isomorphic graphs.

Proposition 14. *There is an isomorphism, up to structural equivalence and injective renaming, between concurrent nets and prefix-closed $\mathcal{C}$-terms with their respective reductions.*

Proof. The canonical form property from Proposition 10 provides the translation between both formalisms: for a $\mathcal{C}$-term P, the associated net $[\![P]\!]$ is $(\mathcal{C}, \mathcal{I}, \mathcal{A}, \vdash)$, where $\mathcal{C}$ is the set of channels (bound or free), $\mathcal{I}$ is the set of free channels, $\mathcal{A}$ is the set of locations $\{\ell_i | 1 \leqslant i \leqslant n\}$ with ℓ_i labelled by $(\varepsilon_i, u_i, \tilde{x}_i)$. The enabling relation is defined as $\ell_{i,j,1}, \ldots, \ell_{i,j,q_{i,j}} \vdash \ell_i$ for each pair (i, j). One easily checks that this translation is a bijection (up to structural equivalence) and that it commutes with reduction, in the sense that there is a translation labelled $P \xrightarrow{\varphi, L} P'$ if and only if $[\![P]\!]$ reduces into $[\![P']\!]$ with equivalence φ. □

As a consequence, in the sequel we use the name *concurrent nets* both for nets in the sense of Definition 11 and for prefix-closed $\mathcal{C}$-terms, and we refer to processes indifferently using the notations for terms or for nets, whichever is the more suitable.

3. Sub-calculi

Several process calculi can be considered as fragments of concurrent nets.

Definition 15. A calculus S is a sub-calculus of $\mathcal{C}$-terms if there is a translation map $t : S \to \mathcal{C}$, modulo the structural equivalences of S and $\mathcal{C}$, that is full and faithful (t is injective on processes and bijective on transitions). A correctness criterion is a characterisation of the image of t.

3.1. Restrictions on the guards

The solos calculus [8] without replication is a sub-calculus of $\mathcal{C}$-terms by the trivial translation $\alpha \mapsto (\nu\ell)(\ell : \alpha)$. It is the fragment of $\mathcal{C}$-terms with no prefixes. Incidentally, solo diagrams [7] as defined by Laneve, Parrow and Victor are very similar to our graphical syntax, since the diagram for a term in the solos calculus is exactly the dual graph of the concurrent net for its translation, in the sense that the vertices and the edges in the diagram are, respectively, the edges and the nodes in the concurrent net.

The solos calculus is a fragment of the fusion calculus [12]. Fusion terms (without replication or sums) are defined by the grammar

$$P := \mathbf{0} \mid (P \mid P) \mid (\nu x)P \mid \bar{u}(\tilde{x}).P \mid u(\tilde{x}).P.$$

Define the translation $[\![\_]\!]_\pi$ by $[\![\alpha \,.\, P]\!]_\pi = (\nu\ell)(\langle\pi\rangle\ell : \alpha | [\![P]\!]_\ell)$, where ℓ is a fresh location, and by homomorphism on all other constructs. The translation $[\![\_]\!]_1$ makes the fusion calculus a sub-calculus of $\mathcal{C}$-terms. The characterisation of the image of this translation requires the formal definition of prefixing:

Definition 16. Let P be a concurrent net.
- The *prefix* of an action α is the set $(\_ \vdash \alpha) := \{X \mid X \vdash \alpha\}$.
- P has a *simple prefixing* if for all α, either $(\_ \vdash \alpha) = \emptyset$ or there is a β such that $(\_ \vdash \alpha) = \{\{\beta\}\}$.
- If P has a simple prefixing, the *prefixing relation* of P is the relation $\leftarrow$ over the actions defined by $\beta \leftarrow \alpha$ when $\beta \vdash \alpha$.

Proposition 17. *The image of the translation of fusion calculus is the set of processes with simple prefixing in which the prefixing relation is acyclic.*

Proof. In this case the arrows form a directed acyclic graph, with nodes of degree at most 1 by hypothesis, i.e. a forest, which corresponds to the syntactical structure of the prefixes in the fusion term. □

A further restriction of prefixing that is found in the literature is that of asynchrony: in ansychronous π-calculus, emissions never act as prefixes, and the translation of this condition in concurrent nets simply states that positive actions never occur on the left of the enabling relation.

3.2. Restrictions on scoping

The π-calculus may be seen as the sub-calculus of the fusion calculus where receptions always appear in the form $(\nu\tilde{x})u(\tilde{x}).P$, where the x_i are distinct, i.e. where receptions are binders. Hence our translation of π-calculus is that of fusion calculus except that restrictions are added before every reception. The characterisation of translations of π-terms requires a formal definition of causality which means that one cannot interact on an unknown channel:

Definition 18. The causality relation is the binary relation $\Leftarrow$ over actions defined by $\beta \Leftarrow \alpha$ if the action β is negative and the principal channel of α or any of its auxiliary channels is an auxiliary channel of β.

Proposition 19. *The image of the translation of π-calculus is the set of processes with a simple prefixing such that the relation $\leftarrow$ is acyclic, the relation $\Leftarrow$ is included in the transitive closure of $\leftarrow$, and the auxiliary channels of negative actions are pairwise distinct.*

Proof. The condition on $\Leftarrow$ imposes that the auxiliary channels of receptions are used only in actions prefixed (possibly indirectly) by this reception, therefore the scope of received channels can always be written as $(\nu\tilde{x})u(\tilde{x}).P$. □

In the definition of $\Leftarrow$, the left action is supposed to be negative, since only negative actions impose scoping in π-calculus. If the definition was extended to a causality $\Leftarrow$ between actions of arbitrary polarity, then the statement of Proposition 19 would capture the fragment known as *private* π-calculus [13].

As the relation $\Leftarrow$ is defined with no reference to prefixing, this privacy condition can be formulated independently, by requiring $\Leftarrow$ to be acyclic, which indeed associates a scope to each negative action; the resulting fragment is to π-calculus what concurrent nets are to fusion calculus, in particular it allows communication to be formulated as name substitution instead of unification, while allowing arbitrary monotonic guards. Requiring $\Leftarrow$ to be acyclic further restricts the calculus into a kind of "private" concurrent nets.

4. Encoding guards

Our notion of guard provides a flexible extension of prefixing that extends fusion calculi with expressive scheduling features. Nevertheless, concurrent nets are still reasonable from the point of view of implementation and algebraic study, since they can be encoded into its fragment without guards, i.e. the solos calculus, as we will see in this section.

The purpose of the enabling relation is to enforce some ordering between the actions of a process, i.e. to restrict the reduction strategy of a $\mathcal{C}$-term. A useful analogy can be drawn with λ-calculus: there are several standard reduction strategies for normalising λ-terms, but the technique of continuation-passing-style transformation provides a way to enforce a particular strategy, modulo a slight modification of the term's interface: each term gets a continuation as an extra argument. The approach we use is similar in that we introduce extra arguments to each action so that communications have the side-effect of connecting other actions that could not interact before.

The idea is to use the properties of fusion to translate explicit delaying of actions into delayed unification of names. The same kind of idea was used by Laneve and Victor to encode fusion calculus into solos [8], but their actual encoding depended on the structure of prefixes. The ones we provide here take a more geometric approach.

4.1. Expansive translation

In this section, we define a translation that we call *expansive* because it introduces a pair of auxiliary ports at each node in the net for each enabling arrow. It implies that the translation fails to be fully compositional: the translation of a parallel composition is not exactly the parallel composition of the translations. As an example, illustrated in Fig. 3, consider we have a concurrent net P of the form

$$P := \ell_1 : \bar{a}(\tilde{x}) \mid \ell_2 : \bar{b}(\tilde{y}) \mid \ell_3 : c(\tilde{z}) \mid \langle \ell_1 \ell_2 + \ell_3 \rangle d(\tilde{w}) \mid Q.$$

This expresses that $d(\tilde{w})$ is blocked until either $c(\tilde{z})$ or both $\bar{a}(\tilde{x})$ and $b(\tilde{y})$ are performed. Thus the way of delaying $d(\tilde{w})$ is to replace d with a fresh channel d' that will be unified with d when this condition is fulfilled. This is achieved by extending the arity of each action, passing a pair of channels to be unified to the actions at locations ℓ_1, ℓ_2 and ℓ_3 and assuming all other actions will perform this unification. Translating the arrow $a(\tilde{x}), \bar{b}(\tilde{y}) \vdash d(\tilde{w})$ leads to

$$P_1 := (\nu d' d_1)\big(\bar{a}(d_1 d' \tilde{x}) \mid \bar{b}(d d_1 \tilde{y}) \mid (\nu x)\ell_3 : c(x x \tilde{z}) \mid \langle \ell_3 \rangle (\nu x) d'(x x \tilde{w})\big) \mid Q_1,$$

where Q_1 is Q where each action $\ell : u^{\varepsilon}(\tilde{v})$ is replaced by $(\nu x)\ell : u^{\varepsilon}(x x \tilde{v})$. Translating the second arrow leads to

$$P_2 := (\nu d' d_1)\big((\nu y)\bar{a}(y y d_1 d' \tilde{x}) \mid (\nu x)\bar{b}(x x d d_1 \tilde{y}) \mid (\nu x) c(d d' x x \tilde{z}) \mid (\nu x y) d'(x x y y \tilde{w})\big) \mid Q_2,$$

where Q_2 is Q_1 transformed in the same way as Q is transformed into Q_1.

Definition 20. Let P be a concurrent net. Assume a particular enumeration of P is chosen. For all i, let u'_i be u_i if $\pi_i \equiv 1$ and a fresh name otherwise. Let $(u_{i,j,k})$ be a family of channel names such that $u_{i,j,0} = u_i$ and $u_{i,j,q_{i,j}} = u'_i$ and all other $u_{i,j,k}$ are fresh. The translation of P for this enumeration is

$$(\nu \tilde{w}') \prod_{I=1}^{n} (\nu \tilde{z}_I) u'_I(\tilde{y}_I \tilde{x}_I) \quad \text{where} \quad y_{I,(i,j,\eta)} = \begin{cases} u_{i,j,k-\eta} & \text{if } \ell_I = \ell_{i,j,k} \text{ for some } k, \\ z_{I,(i,j)} & \text{otherwise,} \end{cases}$$
$$\tilde{w}' = \tilde{w} \cup \{u_{i,j,k} \mid k \neq 0\},$$

where $\tilde{z}_I$ is a family of fresh channels indexed over $\{(i, j) \mid 1 \leqslant i \leqslant n, 1 \leqslant j \leqslant p_i\}$ and $\tilde{y}_I$ is a family of channels indexed over $\{(i, j, \eta) \mid 1 \leqslant i \leqslant n, 1 \leqslant j \leqslant p_i, \eta \in \{1, 0\}\}$. The expansive translation of P is the set $[\![P]\!]_e$ of expansive translations of P for all possible enumerations (thus it should be understood as a relation between processes rather than a function).

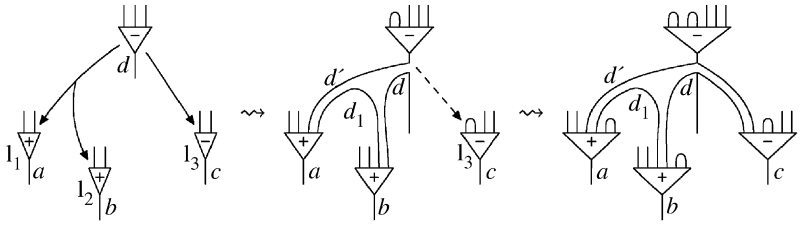


Fig. 3. Expansive translation.

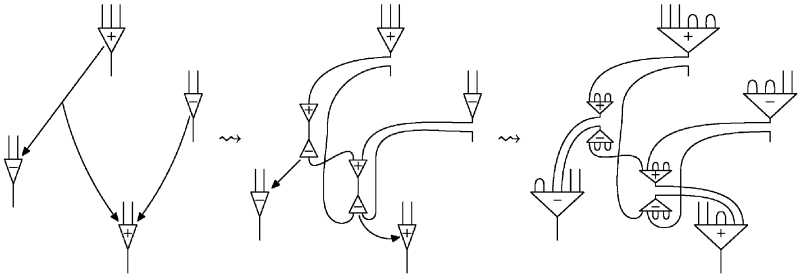


Fig. 4. The two steps of duo translation.

Note that, in this definition, the extra arguments $\tilde{y}_I$ of the actions are indexed on a set of triples, without specifying a precise ordering. It is understood that the ordering is unimportant as long as the same one is used for all actions in a given process.

Theorem 21. *Every process P is stably barb-bisimilar to each $Q \in [\![P]\!]_e$.*

Proof (*Sketch*). The main idea is that the translation commutes with the transition relation, up to the introduction of dummy pairs of arguments in each action, with a few technical details explained in the appendix.

4.2. Duo translation

The translation above has the advantage that it keeps constant the number of nodes and that the translated process is strongly bisimilar to the original one. The major drawback is that the translation is not compositional. We now introduce a translation that addresses this problem by first translating an arbitrary process into one with simpler prefixing, namely a prefixing of depth 1, similar to Parrow's *duos* [11], and then providing a specific translation of this simple prefixing.

As illustrated in the first part of Fig. 4, the first step consists in associating a forwarder $(\boldsymbol{\nu u})(u(\tilde{x})|\bar{u}(\tilde{y}))$ to each node $\ell : \alpha$ that appears in another action's prefix. This forwarder is the one that will delay a fusion, so it is enough to prefix it with $\langle \ell \rangle$ to ensure that the concerned nodes are blocked.

Definition 22. Let P be a concurrent net. Assume a particular enumeration of P is chosen. For all i, let u'_i be u_i if $\pi_i \equiv 1$ and a fresh name otherwise. Let $(u_{i,j,k})$ be a family of channel names such that $u_{i,j,0} = u_i$ and $u_{i,j,q_{i,j}} = u'_i$ and all other $u_{i,j,k}$ are fresh. The duo translation of P for this enumeration is

$$(\boldsymbol{\nu}\tilde{w}') \prod_{I=1}^{n} (\boldsymbol{\nu} v)\big(\ell_I : u'^{\varepsilon_I}_I(\tilde{x}_I) \mid \langle \ell_I \rangle v(\tilde{y}_I) \mid \bar{v}(\tilde{z}_I)\big) \quad \text{with } \tilde{w}' = \tilde{w} \cup \{u_{i,j,k} \mid k \neq 0\},$$

where $\tilde{y}_I$ and $\tilde{z}_I$ are families indexed over $\{(i, j, k) | \ell_{i,j,k} = \ell_I\}$ defined as

$$y_{I,(i,j,k)} = u_{i,j,k-1} \quad \text{and} \quad z_{I,(i,j,k)} = u_{i,j,k}.$$

The duo translation of P is the set $[\![P]\!]_{d1}$ of duo translations of P for all possible enumerations, thus duo translation should be understood as a relation between processes rather than a function.

Theorem 23. *Every process P is weakly stably bisimilar to each $Q \in [\![P]\!]_{d1}$.*

Proof (*Sketch*). The simulation is achieved by reducing all the forwarders introduced by the translation when necessary. These redexes cannot interfere with any other communication, which ensures bisimilarity. The proof is mostly technical, it can be found in the appendix.

Remark that if two processes P and Q are prefix-closed, then $[\![P|Q]\!]_{d1} = [\![P]\!]_{d1}|[\![Q]\!]_{d1}$, which was false for the expansive translation. The advantage of this intermediate form is that it uses a very restricted form of prefixing:

Definition 24. A process P has a duo prefixing if there is a partial injection p over $\{1 \dots n\}$ such that P can be written as

$$P \equiv (\nu \tilde{w}) \prod_{i=1}^{n} \langle \pi_i \rangle \ell_i : u_i^{\varepsilon_i}(\tilde{x}_i) \quad \text{with } \pi_i = \begin{cases} \ell_{p(i)} & \text{if } i \in \mathrm{dom}(p), \\ 1 & \text{otherwise.} \end{cases}$$

Lemma 25. *For every process P, every $Q \in [\![P]\!]_{d1}$ has a duo prefixing.*

The second step of the translation, as illustrated in Fig. 4, consists in encoding the remaining prefixes using the same technique as for the expansive translation from Definition 20. The key difference is that all prefixing arrows are now independent, so we do not need to introduce a pair of extra argument for every arrow: one pair for each polarity is enough.

Definition 26. Let P be a process with a duo prefixing. Assume an enumeration of P with the notations of Definition 24. Let (u'_i) be a family of fresh names indexed over the domain of p. The duo encoding of P for this enumeration is $(\nu \tilde{w} \tilde{u}') \prod_{i=1}^{n} (\nu yz) \ell_i : \alpha_i$ where

$$\alpha_i = \begin{cases} u^{\varepsilon_i}(yyzz\tilde{x}_i) & \text{if } i \notin \mathrm{rng}(p) \\ \bar{u}(yyu_{p^{-1}(i)}u'_{p^{-1}(i)}\tilde{x}_i) & \text{if } \varepsilon_i = + \\ u(u_{p^{-1}(i)}u'_{p^{-1}(i)}zz\tilde{x}_i) & \text{if } \varepsilon_i = - \end{cases} \quad \text{with } u = \begin{cases} u'_i & \text{if } i \in \mathrm{dom}(p), \\ u_i & \text{otherwise.} \end{cases}$$

The duo encoding $[\![P]\!]_{d2}$ is the set of duo encodings of P for all enumerations.

Theorem 27. *Duo encoding is a stable barbed bisimulation.*

Proof (*Sketch*). The actions in a process and in any of its encodings are in one-to-one correspondence. The four extra arguments of each action in the translation simulate precisely the unblocking of locations. The technical details of the proof can be found in the appendix.

Remark again that by construction, for any prefix-closed processes P and Q with duo prefixing, we have $[\![P|Q]\!]_{d2} = [\![P]\!]_{d2}|[\![Q]\!]_{d2}$. As a corollary of Theorems 23 and 27, any process is weakly stably barb-bisimilar to a process without guards, by composition of the encodings, and the composed encoding $[\![-]\!]_d$ commutes with parallel composition of prefix-closed processes.

5. Replication

The calculus we have presented so far does not provide replication nor recursion. In this section, we first extend the definitions with a replication operator, then we show how the results extend to the non-finite case.

The proper way to implement potentially infinite behaviours in a process calculus is lazy replication (or recursion). Replication consists in introducing terms of the form $!P$ that act like as many copies of P as needed by the context.

By *lazy* we mean that duplication is actually performed only on interaction, rather than using a congruence rule like $!P \equiv !P|P$.

Be it lazy or not, duplication in concurrent nets requires some discipline with respect to location names. Indeed, our definition of $\mathcal{C}$-terms requires each location to be defined at most once, and this constraint cannot be preserved by reduction in a replicated process like $!(\ell : \alpha)$, since the name ℓ would occur in every copy of $\ell : \alpha$. One can think of two ways to get around this problem:

- Remove the uniqueness constraint on location names. This would lead to a slightly different theory for the calculus, with a new construct $\ell : P$ that sets the name ℓ to 1 as soon as any action in P is performed. The drawback of this approach is that we loose the very compact algebraic formulation of Definition 11 and the formulation of the encoding theorems gets even more tricky.
- Make the $!P$ construct a binder for all location names. This stresses the fact that a replicated process is a kind of server that waits for communications to be actually activated. In this context, referring to locations inside P from outside $!P$ does not make sense.

The approach we take is the second one: in $!P$, a copy of P is produced whenever an interaction happens, and no locations defined in P are public. This leads to the following rule:

$$\frac{P \xrightarrow{(\nu\tilde{x})\alpha, L} P'}{!P \xrightarrow{(\nu\tilde{x})\alpha, \emptyset} !P \mid P'}.$$

In order to keep a normal form property in the style of Proposition 10, we need two extra structural equivalence rules. The first one represents the fact that ! is a binder for location names and the second one accounts for distribution of prefixes over all copies of a replicated process:

$$!(\nu\ell)P \equiv !P \qquad \text{for each } \ell \in \mathrm{loc}(P)$$
$$\langle\pi\rangle !P \equiv !\langle\pi\rangle P, \qquad \text{if no location occurring in } \pi \text{ is defined in } P.$$

With these rules, we get the following normalisation lemma:

Proposition 28. *Any prefix-closed* $\mathcal{C}$-term P *with replication is structurally equivalent to a term with the following shape, where the product stands for parallel composition*:

$$P \equiv (\nu\tilde{w})(\nu\tilde{m})\left(\prod_{i=1}^{n} \langle\pi_i\rangle \ell_i : u_i^{\varepsilon_i}(\tilde{x}_i) \,\middle|\, \prod_{i=1}^{b} !P_i\right) \quad \text{with } \pi_i \equiv \sum_{j=1}^{p_i} \ell_{i,j,1} \cdots \ell_{i,j,q_{i,j}}$$

for some $b \geqslant 0$, $n \geqslant 0$, $p_i \geqslant 0$ *and* $q_{i,j} \geqslant 0$, *where the* $\ell_{i,j,k}$ *are elements of* $\{\ell_i \mid 1 \leqslant i \leqslant n\}$. *The sets* $\tilde{w}$ *and* $\tilde{m}$ *represent, respectively, private channels and locations. Such a formulation is called an* enumeration *of* P.

When the P_i are also prefix-closed, they may in turn be enumerated the same way. In that case, we can extend the duo translation to processes with replication by simply translating the replicated processes:

Definition 29. The duo translation of a prefix-closed process with prefix-closed replications is defined inductively as

$$\left[\!\left[P_0 \,\middle|\, \prod_{i=0}^{b} !P_i \right]\!\right]_d := [\![P_0]\!]_d \,\middle|\, \prod_{i=0}^{b} !\,[\![P_i]\!]_d,$$

where $[\![P_0]\!]_d$ is the translation as specified in Definitions 22 and 26.

Theorem 30. *Any prefix-closed process* P *with prefix-closed replications is weakly stably barb-bisimilar to each* $Q \in [\![P]\!]_d$.

Proof (*Sketch*). We use the notations of Definition 29. Since each P_i is prefix-closed, for any integer k_i we have $[\![P_i^{k_i}]\!]_d = [\![P_i]\!]_d^{k_i}$ by compositionality, so if we denote by P_0 the non-replicated part, we have

$$\left[\!\left[P_0 \,\middle|\, \prod_{i=1}^{b} P_i^{k_i} \right]\!\right]_d = [\![P_0]\!]_d \,\middle|\, \prod_{i=1}^{b} [\![P_i]\!]_d^{k_i}$$

from which we can deduce that

$$P_0 \left| \prod_{i=0}^{b} !P_i \quad \text{and} \quad [\![P_0]\!]_d \right| \prod_{i=0}^{b} ![\![P_i]\!]_d$$

are weakly stably barb-bisimilar. The precise proof of this fact consists in building a bisimulation that relates each process of the form $P_0 \mid \prod_{i=1}^{b} !P_i \mid P_i^{k_i}$ with $k_i \in \mathbb{N}$ to all the processes of the form $Q_0 \mid \prod_{i=1}^{b} (!Q_{i,0} \mid Q_{i,1} \mid \cdots \mid Q_{i,k_i'})$ with $Q_0 \in [\![P_0]\!]_d$ and $Q_{i,j} \in [\![P_i]\!]_d$ for each i and j. This way copies of the replicated sub-processes can be introduced while remaining inside the bisimulation.

The natural question then is to ask whether the result still holds when the replicated sub-processes are not prefix-closed, that is, when a replicated action may be prefixed by a non-replicated one. It is interesting to remark that the translations defined above actually fail in this case. As an example, consider the following process:

$$P := \ell : \bar{a}(b) \mid !(\boldsymbol{\nu} x)\big(c(x) \mid \langle \ell \rangle \bar{x}(y)\big).$$

The duo translation as defined in Definition 22 cannot be applied here, since it would require extending the scope of x to ℓ, and of course no scope extrusion allows extending a scope from inside a replication. This stresses the fact that prefixing is actually a form of communication, with the same notions of privacy and scoping as those well known for channel names in communications. Nevertheless, the process P above can be translated into a weakly bisimilar process where the prefixing of ℓ moves to an action on a public channel:

$$\begin{aligned} P' &:= (\boldsymbol{\nu} uv)\big(\ell : \bar{a}(b) \mid \langle \ell \rangle \{u = v\} \mid !Q\big), \\ \text{with } Q &:= (\boldsymbol{\nu} xx')\big(m : c(x) \mid \bar{x}'(y) \mid \langle m \rangle \bar{u}(xx') \mid (\boldsymbol{\nu} z)v(zz)\big), \end{aligned}$$

where $\{u = v\}$ is a process that fuses u and v, for instance $(\boldsymbol{\nu} x)(\bar{x}(u) | x(v))$. The point here is that the names x and x' remain private, and their fusion is done by a communication between the public channels u and v once they are unified. The role of the prefixing in $\langle m \rangle \bar{u}(xx')$ is to forbid communication between copies of Q and $!Q$, because such communications would trigger new copies of Q, making the process diverge, which would break bisimilarity. The guard is m because this is the only public action in Q (apart from the ones on u and v of course), the point is that it is unblocked when the copy of Q is made. Communications on u between different copies of Q is not a problem since they cannot create interference, by construction.

The previous example illustrates how processes with replication and arbitrary prefixing may be translated into prefix-less ones. We will not embark in the definition of a precise translation of arbitrary processes into processes with prefix-closed replications, because a formal definition would provide hardly any new insight with respect to the example above.

6. Conclusions and future works

The calculus of concurrent nets extends the family of π-like calculi by introducing a new and more expressive form of sequentiality constraints. Prefixing is seen as the evolution of a shared state, in a way that recalls synchronisation mechanisms like semaphores. In its graphical form, the calculus makes clear the notions of locality and scoping and expresses prefixing as a form of interaction with a geometric intuition. The graphical presentation distinguishes connectivity (of communication channels) and scheduling, which can be considered as locality. This distinction is similar to that at the origin of the approach of bigraphs [5], a more precise study of the relationships between both formalisms could provide interesting insights.

We show, by means of encodings, that in a sense the calculus without guards has the same expressive power as the calculus with guards. Hence, while keeping the same theoretical expressiveness, concurrent nets provide new programming features for concurrent calculi, as well as a clearer separation between the geometric part of communication and the scheduling.

The graphical presentation of the calculus, apart from providing strong visual intuitions on the structure of processes, has the advantage of being algebraically simpler and needing almost no structural congruence. The equality (or graph isomorphism) of algebraic definitions of processes seems close to capturing strong bisimulation, hence it is an interesting

step towards the definition of a proper semantics for concurrent nets. The structural rules we have for prefixes suggest a semantics that would make guards and scheduling constraints a first-order notion with interesting properties.

While the use of an enabling relation is a departure from syntactical prefixing, the treatment of replication is still directly inherited from term syntaxes. A similar method could be applied to this feature of the calculus, for instance, by developing a graphical formalism for shared copying, possibly in the style of interaction nets [6]. Those are a notable example of a concurrent (or at least parallel) graphical calculus, with some similar intuitions and several features that make them a quite different model. They are related to the formalism of proof-nets in linear logic [4], and these objects were actually a source of inspiration in this work. Using appropriate interpretations, proof-nets may indeed be considered as a very particular sub-calculus of concurrent nets. This will be detailed in a future paper.

The prefixes in the term calculus are arbitrary monotonic functions, built by conjunction and disjunction on monotonic variables. What makes their encoding possible is the fact that communication by fusion has the same monotonic flavour. The extension to non-monotonic guards, for instance, by introducing negation, would strictly extend the expressiveness, actually it seems to be related to the introduction of external choice in the calculus. Searching further, the mechanism of guards seems general enough to introduce other forms of control, like timed transitions or probabilistic choice, although these ideas remain to be explored.

Appendix A. Proofs

In this appendix, we detail the technical proofs that we did not include in the main text. We do not provide a complete proof for Theorem 21 (strong barbed bisimulation of the expansive translation) as it is essentially the same as Theorem 27 (strong barbed bisimulation for duo encoding). The differences are explained in Section A.3 at the end.

A.1. Duo translation

Note that many enumerations of a given process exist, since these depend on the order of the actions and the formulation of the guards. In particular, a guard $\pi = 1$ may as well be formulated $\pi = 1 + \pi'$ though these are structurally equivalent.

Definition 31. Let $\tilde{x}$ and $\tilde{y}$ be two sequences of channel names with $|\tilde{x}| = |\tilde{y}|$. The delayed fusion of $\tilde{x}$ and $\tilde{y}$ is $F_{\tilde{x}=\tilde{y}} := (\nu u)(u(\tilde{x})|\bar{u}(\tilde{y}))$. A set of delayed fusions of type φ is a parallel composition of delayed fusions $F_\varphi = F_{\tilde{x}_1=\tilde{y}_1} \mid \cdots \mid F_{\tilde{x}_n=\tilde{y}_n}$ such that φ is the equivalence $\tilde{x}_1 \cdots \tilde{x}_n = \tilde{y}_1 \cdots \tilde{y}_n$.

Lemma 32. *For any delayed fusion,* $F_{\tilde{x}=\tilde{y}} \xrightarrow{\{\tilde{x}=\tilde{y}\},\emptyset} \mathbf{0}$.

Proof. Obvious from the definition of the transition system. □

Definition 33. Let $\mathcal{A}$ be a set of processes. The fusion expansion of $\mathcal{A}$ is the set $\mathcal{A}^{\exp} = \left\{(\nu\tilde{x})(F_\varphi|P') \;\middle|\; \mathrm{dom}(\varphi) \subset \tilde{x}, \forall\sigma : \varphi, \exists P \in \mathcal{A}, (\nu\tilde{x})(P'\sigma) \equiv P\right\}$, where $\forall\sigma : \varphi$ means "for all substitution σ that implements φ."

Lemma 34. *For any process* P, *for all* $P' \in \{P\}^{\exp}$, $P' \to^* P$.

Proof. By Lemma 32, the transitions of F_φ produce the effect of φ within the scope of $\tilde{x}$, and they are τ-transitions since all names that are affected by φ are hidden in P' by hypothesis. □

Proof of Theorem 23. Let $\mathcal{S}$ be the relation over processes that relates each P to all the fusion expansions of its duo translations, i.e. $P\mathcal{S}Q$ for each $Q \in [\![P]\!]^{\exp}_{d1}$. We prove that $\mathcal{S}$ and $\mathcal{S}^{-1}$ are weak simulations.

Let (P, Q) be a pair of processes related by $\mathcal{S}$. By definition, we have $Q = (\nu\tilde{w}')(F_\chi|R)$ and for any substitution σ that implements χ, $(\nu\tilde{w}')R\sigma$ is structurally equivalent to some duo translation of P. Pick such a substitution σ. Using

the notations of Proposition 10 and Definition 22, there is an enumeration of P such that

$$(\nu\tilde{w}')R\sigma \equiv \prod_{i=1}^{n} A_i \quad \text{with } A_i = (\nu v)\big(\ell_i : u_i'^{\varepsilon_i}(\tilde{x}_i) \mid \langle \ell_i \rangle v(\tilde{y}_i) \mid \bar{v}(\tilde{z}_i)\big).$$

In the following, for all i we write $\alpha_i = u_i^{\varepsilon_i}(\tilde{x}_i)$ in order to simplify the expression of the enumeration of P.

If a transition $P \xrightarrow{\varphi, L} P'$ exists, then there are two indices a and b such that $\varphi = \{\tilde{x}_a = \tilde{x}_b\} \setminus \tilde{w}$ and $L = \{\ell_a, \ell_b\}$ and the derivation of the transition can be written

$$\dfrac{\dfrac{\dfrac{\overline{\ell_a : \alpha_a \xrightarrow{\alpha_a, \{\ell_a\}} \mathbf{0}} \qquad \overline{\ell_b : \alpha_b \xrightarrow{\alpha_b, \{\ell_b\}} \mathbf{0}}}{\ell_a : \alpha_a \mid \ell_b : \alpha_b \xrightarrow{\{\tilde{x}_a = \tilde{x}_b\}, L} \mathbf{0}}}{\prod_{i=1}^{n} \langle \pi_i \rangle \ell_i : \alpha_i \xrightarrow{\{\tilde{x}_a = \tilde{x}_b\}, L} \prod_{i \notin \{a,b\}} \langle \pi_i[1/L] \rangle \ell_i : \alpha_i}}{P \xrightarrow{\varphi, L} (\nu\tilde{w}) \prod_{i \notin \{a,b\}} \langle \pi_i[1/L] \rangle \ell_i : \alpha_i}$$

and the right-hand side of the last rule is structurally equivalent to P'. After the substitution $1/L$, the family $(\ell_{i,j,k})$ can be reindexed as $(m_{i,j,k})$ with $1 \leqslant i \leqslant n$, $1 \leqslant j \leqslant p_i$ and $1 \leqslant k \leqslant q'_{i,j}$ where $q'_{i,j}$ is $q_{i,j}$ minus the number of occurrences of ℓ_a or ℓ_b in $\ell_{i,j,1} \ldots \ell_{i,j,q_{i,j}}$. This way we have for each i

$$\pi_i[1/L] = \left(\sum_{j=1}^{p_i} \ell_{i,j,1} \cdots \ell_{i,j,q_{i,j}} \right)[1/L] = \sum_{j=1}^{p_i} m_{i,j,1} \cdots m_{i,j,q'_{i,j}},$$

which yields an enumeration of P'.

Since the actions at ℓ_a and ℓ_b are enabled, for $i \in \{a, b\}$ we have $u'_i = u_i$ and the following holds:

$$\dfrac{\dfrac{\overline{\ell_i : \alpha_i . v(\tilde{y}_i) \xrightarrow{\alpha_i, \{\ell_i\}} v(\tilde{y}_i)}}{\ell_i : \alpha_i \mid \langle \ell_i \rangle v(\tilde{y}_i) \mid \bar{v}(\tilde{z}_i) \xrightarrow{\alpha_i, \{\ell_i\}} v(\tilde{y}_i) \mid \bar{v}(\tilde{z}_i)}}{A_i \xrightarrow{\alpha_i, \{\ell_i\}} (\nu v)(v(\tilde{y}_i) \mid \bar{v}(\tilde{z}_i))},$$

and using the notation of Definition 31, with $\psi = \{\tilde{y}_a = \tilde{z}_a, \tilde{y}_b = \tilde{z}_b\}$ and $F_\psi = F_{\tilde{y}_a = \tilde{z}_a} | F_{\tilde{y}_b = \tilde{z}_b}$:

$$\dfrac{\dfrac{\dfrac{A_a \xrightarrow{\alpha_a, \{\ell_a\}} F_{\tilde{y}_a = \tilde{z}_a} \qquad A_b \xrightarrow{\alpha_b, \{\ell_b\}} F_{\tilde{y}_b = \tilde{z}_b}}{A_a \mid A_b \xrightarrow{\{\tilde{x}_a = \tilde{x}_b\}, L} F_\psi}}{R\sigma \equiv \prod_{j=1}^{n} A_j \xrightarrow{\{\tilde{x}_a = \tilde{x}_b\}, L} F_\psi \mid \prod_{i \notin \{a,b\}} A_i}}{(\nu\tilde{w}')R\sigma \xrightarrow{\varphi, L} (\nu\tilde{w}') \left(F_\psi \mid \prod_{i \notin \{a,b\}} A_i \right)}.$$

Note that the substitution $1/L$ is not necessary in the right-hand sides since the locations ℓ_a and ℓ_b do not occur in F_ψ nor in any A_i with $i \notin \{a, b\}$, by construction. Call $R' = (\nu\tilde{w}')(F_\psi | Q')$ the reduct in the last line. By definition, $\tilde{y}_a$ and $\tilde{z}_a$ are families indexed over $\{(i, j, k) | \ell_{i,j,k} = \ell_a\}$ defined as

$$y_{a,(i,j,k)} = u_{i,j,k-1} \quad \text{and} \quad z_{a,(i,j,k)} = u_{i,j,k},$$

so the fusion $\tilde{y}_a = \tilde{z}_a$ unifies each pair $(u_{i,j,k-1}, u_{i,j,k})$ such that $\ell_{i,j,k} = \ell_a$. The same goes for $\tilde{y}_b$ and $\tilde{z}_b$. By Lemma 34, if τ is a substitution that implements $\psi = \{\tilde{y}_a = \tilde{z}_a, \tilde{y}_b = \tilde{z}_b\}$, we have

$$R' = (\nu\tilde{w}') \left(F_\psi \,\middle|\, \prod_{i \notin \{a,b\}} A_i \right) \to^* (\nu\tilde{w}') \prod_{i \notin \{a,b\}} A_i \tau. \tag{A.1}$$

Besides, $(u_{i,j,k}\tau)$ can be reindexed as $(v_{i,j,k})$ with the same index transformation that transformed $(\ell_{i,j,k})$ into $(m_{i,j,k})$. Define v'_i to be u_i if $\pi_i[1/L] \equiv 1$ and u'_i otherwise. With these notations, we get for each $i \notin \{a, b\}$

$$A_i\tau = (\nu w)\big(\ell_i : v'^{\varepsilon_i}_i(\tilde{x}_i\tau) \mid \langle\ell_i\rangle w(\tilde{y}'_i) \mid \bar{w}(\tilde{z}'_i)\big).$$

Then the process $(\nu\tilde{w}')\prod_{i\notin\{a,b\}} A_i\tau$ is the duo translation of P' for the enumeration we get by reindexing. By Eq. (A.1), we deduce that $R' \in [\![P']\!]^{\exp}_{d1}$, and since $Q \to^* R\sigma \to R'$ we have that $\mathcal{S}$ is a weak simulation.

We now have to prove that $\mathcal{S}^{-1}$ is also a weak simulation. Again, consider a pair (P, Q) in $\mathcal{S}$, with $Q = (\nu\tilde{w}')(F_\chi|R)$, and where R is decomposed as

$$R \equiv \prod_{i=1}^{n} A_i \quad \text{with } A_i = (\nu v)\big(\ell_i : u'^{\varepsilon_i}_i(\tilde{x}_i) \mid \langle\ell_i\rangle v(\tilde{y}_i) \mid \bar{v}(\tilde{z}_i)\big).$$

If a reduction $Q \xrightarrow{\varphi,L} Q'$ exists, it affects either a redex in F_χ or a redex in R, since the subjects of actions in F_χ are each shared between exactly two opposite actions.

If the reduction happens in F_χ, it concerns a delayed fusion $F_{\tilde{x}=\tilde{y}}$ and we have $Q' = (\nu\tilde{w}')(F_{\chi'}|R)\sigma$, where σ is a substitution that implements $\tilde{x} = \tilde{y}$ and χ' is the equivalence generated by what remains of F_χ. Therefore both Q and Q' are in the fusion expansion of $[\![P]\!]_{d1}$, which means that $P\mathcal{S}Q'$.

If the reduction happens in R, there exist a and b such that u'_a and u'_b are the same channel u, $\varepsilon_a = +$, $\varepsilon_b = -$ and thus $\varphi = \{\tilde{x}_a = \tilde{x}_b\} \setminus \tilde{w}'$ and $L = \{\ell_a, \ell_b\}$. Then if we write $\psi = \{\tilde{y}_a = \tilde{z}_a, \tilde{y}_b = \tilde{z}_b\}$ and $F_\psi = F_{\tilde{y}_a=\tilde{z}_a}|F_{\tilde{y}_b=\tilde{z}_b}$, the derivation of the reduction can be written as

$$\dfrac{\dfrac{\dfrac{A_a \xrightarrow{u(\tilde{x}_a),\{\ell_a\}} F_{\tilde{y}_a=\tilde{z}_a} \qquad A_b \xrightarrow{\bar{u}(\tilde{x}_b),\{\ell_b\}} F_{\tilde{y}_b=\tilde{z}_b}}{A_a \mid A_b \xrightarrow{\{\tilde{x}_a=\tilde{x}_b\},L} F_\psi}}{R \xrightarrow{\{\tilde{x}_a=\tilde{x}_b\},L} F_\psi \mid \prod_{i\notin\{a,b\}} A_i}}{Q \xrightarrow{\varphi,L} (\nu\tilde{w}')\big(F_\chi \mid F_\psi \mid \prod_{i\notin\{a,b\}} A_i\big)\tau \equiv Q'}$$

for some substitution τ that implements $\{\tilde{x}_a = \tilde{x}_b\} \setminus \tilde{w}'$. By definition of the translation, the names u'_i in a duo translation are fresh and pairwise distinct except for those that are equal to some u_i, i.e. those for which $\pi_i \equiv 1$. If σ is a substitution that implements χ, $(\nu\tilde{w}')R\sigma$ is a duo translation of P, so $u'_a\sigma = u'_b\sigma$ does imply that $\pi_a \equiv \pi_b \equiv 1$, and the actions at locations ℓ_a and ℓ_b in P are enabled. As a consequence, the following reduction holds:

$$\dfrac{\dfrac{\dfrac{\ell_a : u_a(\tilde{x}_a) \xrightarrow{u(\tilde{x}_a),\{\ell_a\}} \mathbf{0} \qquad \ell_b : \bar{u}_b(\tilde{x}_b) \xrightarrow{\bar{u}(\tilde{x}_b),\{\ell_b\}} \mathbf{0}}{\ell_a : \alpha_a \mid \ell_b : \alpha_b \xrightarrow{\{\tilde{x}_a=\tilde{x}_b\},\{\ell_a,\ell_b\}} \mathbf{0}}}{\prod_{i=1}^{n}\langle\pi_i\rangle\ell_i : \alpha_i \xrightarrow{\{\tilde{x}_a=\tilde{x}_b\},\{\ell_a,\ell_b\}} \prod_{i\notin\{a,b\}}\langle\pi_i[1/\ell_a,\ell_b]\rangle\ell_i : \alpha_i}}{P \xrightarrow{\varphi,L} \prod_{i\notin\{a,b\}}\langle\pi_i[1/L]\rangle\ell_i : \alpha_i\tau \equiv P'}$$

with the same substitution τ as above. It can be verified, with the same arguments as in the first part, that Q' is a fusion expansion of a duo translation of the reduct P', and therefore we have $P \to P'$ and $P'\mathcal{S}Q'$.

This proves that $\mathcal{S}^{-1}$ is also a weak simulation, therefore the symmetric closure $\mathcal{S} \cup \mathcal{S}^{-1}$ is a weak bisimulation.

The set of barbs (u, L) such that $P\downarrow(u, L)$ is the set of all $(u_i, \{\ell_i\})$ such that $\pi_i \equiv 1$ and $u_i \notin \tilde{w}$. The set of (u, L) such that $(\nu\tilde{w}')R\sigma\downarrow(u, L)$ is the set of $(u'_i, \{\ell_i\})$ where u'_i does not appear in $\tilde{w}'$, that is by definition when u'_i is some u_j that does not appear in $\tilde{w}$. Therefore, $P\downarrow(u, L)$ if and only if $(\nu\tilde{w}')R\sigma\downarrow(u, L)$. Moreover, by definition of the fusion expansion, the free channels in Q are those in $(\nu\tilde{w}')R\sigma$, so $(\nu\tilde{w}')R\sigma\downarrow(u, L)$ if and only if $Q \to^* \downarrow(u, L)$. Therefore, $\mathcal{S}$ is a weak barbed simulation.

Moreover, it is clear from the definition of the translation that it commutes with arbitrary name substitution since the free names in a process and its translations are the same. Since all locations are supposed private in the definition of the duo translation, the condition on substitution of free locations by 1 in the definition of stable bisimulation is empty, so we have a weak stable bisimulation, which concludes the proof of Theorem 23. □

A.2. Duo encoding

Proof of Theorem 27. Let $\mathcal{S}$ be the relation over processes that relates each P to every element of $[\![P]\!]_{d2}$. Let (P, Q) be an element of $\mathcal{S}$. We use the notations of Definitions 24 and 26.

If a transition $P \xrightarrow{\varphi, L} P'$ exists, then there are two indices a and b such that u_a and u_b are the same channel u, $\varphi = \{\tilde{x}_a = \tilde{x}_b\} \setminus \tilde{w}$ and $L = \{\ell_a, \ell_b\}$ and the derivation of the transition can be written as

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\ell_a : u(\tilde{x}_a) \xrightarrow{u(\tilde{x}_a),\{\ell_a\}} \mathbf{0}} \quad \dfrac{}{\ell_b : \bar{u}(\tilde{x}_b) \xrightarrow{\bar{u}(\tilde{x}_b),\{\ell_b\}} \mathbf{0}}}{\ell_a : u(\tilde{x}_a) \mid \ell_b : \bar{u}(\tilde{x}_b) \xrightarrow{\{\tilde{x}_a = \tilde{x}_b\}, L} \mathbf{0}}}{\prod_{i=1}^{n} \langle \pi_i \rangle \ell_i : \alpha_i \xrightarrow{\{\tilde{x}_a = \tilde{x}_b\}, L} \prod_{i \notin \{a,b\}} \langle \pi_i [1/L] \rangle \ell_i : \alpha_i}}{P \xrightarrow{\varphi, L} (\nu \tilde{w}) \prod_{i \notin \{a,b\}} \langle \pi_i [1/L] \rangle \ell_i : \alpha_i}$$

and the right-hand side of the last rule is structurally equivalent to P'. Because of the form of the guards π_i, the substitution $1/L$ may affect at most two guards by replacing them by 1.

The form of the corresponding transition in Q depends on whether a and b are in the range of the partial injection p, which means that four cases have to be considered. The symptomatic case is when both a and b are in the range of p, the other cases are just simpler.

Let a' and b' be the indices such that $a = p(a')$ and $b = p(b')$. Since the actions at ℓ_a and ℓ_b are enabled in P, $\pi_a = \pi_b = 1$ so a' and b' are distinct from a and b. Therefore Q can be written as

$$Q \equiv (\nu \tilde{w} \tilde{u}') \left((\nu z) \ell_a : u(u_{a'} u'_{a'} z z \tilde{x}_a) \,\middle|\, (\nu y) \ell_b : u(y y u_{b'} u'_{b'} \tilde{x}_b) \,\middle|\, \prod_{i \notin \{a,b\}} A_i \right).$$

Writing A_a and A_b for the first two actions, the following reduction holds:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{A_a \xrightarrow{(\nu z) u(u_{a'} u'_{a'} z z \tilde{x}_a),\{\ell_a\}} \mathbf{0}} \quad \dfrac{}{A_b \xrightarrow{(\nu y) u(y y u_{b'} u'_{b'} \tilde{x}_b),\{\ell_b\}} \mathbf{0}}}{A_a \mid A_b \xrightarrow{\{u_{a'} = u'_{a'}, u_{b'} = u'_{b'}, \tilde{x}_a = \tilde{x}_b\}, L} \mathbf{0}}}{\prod_{i=1}^{n} A_i \xrightarrow{\{u_{a'} = u'_{a'}, u_{b'} = u'_{b'}, \tilde{x}_a = \tilde{x}_b\}, L} \prod_{i \notin \{a,b\}} A_i}}{(\nu \tilde{u}') \prod_{i=1}^{n} A_i \xrightarrow{\{\tilde{x}_a = \tilde{x}_b\}, L} (\nu \tilde{u}') \prod_{i \notin \{a,b\}} A_i \sigma}}{Q \xrightarrow{\varphi, L} (\nu \tilde{w} \tilde{u}') \prod_{i \notin \{a,b\}} A_i \sigma}$$

with $\sigma = [u_{a'}/u'_{a'}, u_{b'}/u'_{b'}]$. Call Q' the reduct in the last rule. The substitution σ affects only the actions at indices a' and b' since the channels $u'_{a'}$ and $u'_{b'}$ appear only in these as subjects, and appear only in A_a and A_b as objects, by injectivity of p. Substituting $u_{a'}$ for $u'_{a'}$ in the encoding of P corresponds to substituting 1 for $\pi_{a'}$ in P, i.e. the effect of σ in Q' is exactly the encoding of the effect of $1/L$ in P', therefore we have $Q \xrightarrow{\varphi, L} Q'$ with $P' \mathcal{S} Q'$.

In the other cases, where at most one of a and b is in the range of p, the proof of simulation is similar, and as a consequence $\mathcal{S}$ is a simulation.

If a transition $Q \xrightarrow{\varphi, L} Q'$ exists, it affects a pair of actions of indices a and b with the same subject. By construction, this means that this subject is $u_a = u_b$ since all u'_i are fresh and used exactly once as subjects, and thus a and b are not in the domain of p, so the actions at indices a and b in P are enabled. Then it is easy to check that the reduction of this redex in P leads to a transition $P \xrightarrow{\varphi, L} P'$ with the same label, and by the same arguments as above Q' is a duo encoding of P', so $P' \mathcal{S} Q'$. This proves that $\mathcal{S}^{-1}$ is also a strong bisimulation, so the reflexive closure $\mathcal{S} \cup \mathcal{S}^{-1}$ is a strong bisimulation.

Since the public names and enabled actions in P and Q are the same, it is clear that $P \downarrow (u, L)$ if and only if $Q \downarrow (u, L)$ and that $\mathcal{S}$ is closed under arbitrary name substitution, so we have a strong stable bisimulation. □

A.3. Expansive translation

The definition of expansive translation is similar to that of duo encoding. The main difference is that the expansive translation adds a pair of arguments to each action for each arrow, while the duo encoding adds four arguments in any case.

The relation that is used to prove bisimulation in this case is the $\mathcal{S}$ that relates each process P with the set of all its possible expansive translations (for all enumerations) closed under argument expansion. What we call argument expansion is the following: consider a process P with an enumeration as

$$P = \prod_{i=1}^{n} \langle \pi_i \rangle \ell_i : u_i(\tilde{x}_i)$$

and assume there is a k such that $|\tilde{x}_i| \geqslant k$ for all i. Then the argument expansion at position k for this enumeration is the process

$$\prod_{i=1}^{n} \langle \pi_i \rangle (\boldsymbol{\nu} y y) \ell_i : u_i(x_{i,1} \dots x_{i,k}\, y y\, x_{i,k+1} \dots x_{i,|\tilde{x}_i|}),$$

that is, we add a fresh name twice at the same position in each action. This accounts for the extra arguments in the expansive translation that may be produced during the reduction of a translated process. It is clear that any argument expansion of a process P is strongly barbed-bisimilar to P.

Up to this argument expansion, the bisimulation proof is mostly the same as for duo encoding, so we do not write it in detail. The main difference is that two actions that occur in the same prefix may interact. For instance, consider the process

$$P = \ell : a(x) \mid m : \bar{a}(y) \mid n : b(z) \mid \langle \ell m n \rangle c(t).$$

A possible expansive translation of P is

$$P' = (\boldsymbol{\nu} c_1 c_2 c_3)\big(a(c c_1 x) \mid \bar{a}(c_1 c_2 y) \mid b(c_2 c_3 z) \mid (\boldsymbol{\nu} x) c_3(x x t)\big).$$

In P there is a possible interaction between the actions at locations ℓ and m, which is written in P' as

$$P' \xrightarrow{\{x=y\},\emptyset} (\boldsymbol{\nu} c_3)\big(b(c c_3 z) \mid (\boldsymbol{\nu} x) c_3(x x t)\big)$$

with the internal fusion $c = c_1 = c_2$. As expected, the reduct is a translation of $n : b(z) \mid \langle n \rangle c(t)$, which is a reduct of P by the same transition. One easily checks that such a situation is always correctly handled in the expansive translation.

References

[1] C. Fournet, G. Gonthier, The reflexive CHAM and the join-calculus, in: Proc. POPL'96, ACM Press, New York, 1996, pp. 372–385.

[2] Y. Fu, A proof-theoretical approach to communication, in: P. Degano, R. Gorrieri, A. Marchetti-Spaccamela (Eds.), Proc. ICALP'97, Lecture Notes in Computer Science, Vol. 1256, Springer, Berlin, 1997, pp. 325–335.

[3] P. Gardner, L. Wischik, Explicit fusions, in: M. Nielsen, B. Rovan (Eds.), Proc. MFCS 2000, Lecture Notes in Computer Science, Vol. 1893, Springer, Berlin, 2000, pp. 373–382.

[4] J.-Y. Girard, Proof-nets: the parallel syntax for proof-theory, in: P. Agliano, A. Ursini (Eds.), Logic and Algebra, Marcel Dekker, New York, 1996.

[5] O.H. Jensen, R. Milner, Bigraphs and transitions, ACM SIGPLAN Notices 38 (1) (2003) 38–49.

[6] Y. Lafont, Interaction nets, in: Proc. POPL'90, ACM Press, New York, 1990, pp. 95–108.

[7] C. Laneve, J. Parrow, B. Victor, Solo diagrams, in: N. Kobayashi, B.C. Pierce (Eds.), Proc. TACS'01, Lecture Notes in Computer Science, Vol. 2215, Springer, Berlin, 2001, pp. 127–144.

[8] C. Laneve, B. Victor, Solos in concert, in: J. Wiederman, P. van Emde Boas, M. Nielsen (Eds.), Proc. ICALP'99, Lecture Notes in Computer Science, Vol. 1644, Springer, Berlin, 1999, pp. 513–523.

[9] R. Milner, Pi-nets: a graphical form of pi-calculus, in: D. Sannella (Ed.), Proc. ESOP'94, Lecture Notes in Computer Science, Vol. 788, Springer, Berlin, 1994, pp. 26–42.

[10] R. Milner, J. Parrow, D. Walker, A calculus of mobile processes (Parts I and II), Inform. Comput. 100 (1992) 1–77.

[11] J. Parrow, Trios in concert, in: G. Plotkin, C. Stirling, M. Tofte (Eds.), Proof, Language and Interaction: Essays in Honour of Robin Milner, MIT Press, Cambridge, MA, 1998, pp. 621–637.

[12] J. Parrow, B. Victor, The fusion calculus: expressiveness and symmetry in mobile processes, in: Proc. LICS'98, 1998, pp. 176–185.

[13] D. Sangiorgi, D. Walker, Pi-Calculus: A Theory of Mobile Processes, Cambridge University Press, Cambridge, 2001.